Online Kernel-Based Nonlinear Neyman-Pearson Classification

Basarbatu Can
Faculty of Engineering & Natural Sciences
Sabancı University
Istanbul, Turkey
basarbatucan@sabanciuniv.edu

Mine Kerpicci
The School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, USA
mkerpicci3@gatech.edu

Huseyin Ozkan
Faculty of Engineering & Natural Sciences
Sabancı University
Istanbul, Turkey
huseyin.ozkan@sabanciuniv.edu

***Abstract*—We propose a novel Neyman-Pearson (NP) classification algorithm, which achieves the maximum detection rate and meanwhile keeps the false alarm rate around a user-specified threshold. The proposed method processes data in an online framework with nonlinear modeling capabilities by transforming the observations into a high dimensional space via the random Fourier features. After this transformation, we use a linear classifier whose parameters are sequentially learned. We emphasize that our algorithm is the first online Neyman-Pearson classifier in the literature, which is suitable for both linearly and nonlinearly separable datasets. In our experiments, we investigate the performance of our algorithm on well-known datasets and observe that the proposed online algorithm successfully learns the nonlinear class separations (by outperforming the linear models) while matching the desired false alarm rate.**

***Index Terms*—Neyman-Pearson, online learning, nonlinear classification, kernel, random projections, Fourier features, perceptron.**

I. Introduction

In applications such as disease diagnosis and fraud detection, the available data from the target (anomaly) and non-target (normal) distributions are irregular because of the rarity of the abnormal instances. Moreover, classification errors might well have different consequences depending on the label of the misclassified instance [1], [2]. For instance, in the context of fraud detection, frequent fraud alarm generation results in unnecessary measures to be taken, which eventually decreases the reliability of the predictor. Therefore, it is often of special importance in such applications to track the false alarm rate (Type-I error) of the predictor to ensure reliability. This can be achieved by introducing asymmetric class costs into the binary classification problem, resulting in an equivalent Neyman-Pearson (NP) formulation. Namely, in this formulation, the Type-I error (false alarm rate) is kept below the level set by the user, while the Type-II error (failure to determine the target) is minimized [3].

This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under Contract 118E268.

The state-of-the-art NP classification methods can be divided into three main categories: thresholding [4], [5], cost sensitive learning [6], [7] and constrained optimization [8]. In the thresholding approach, a classifier is trained on a set of data, where the threshold is adjusted accordingly to satisfy the desired false alarm rate. One of the early examples of this approach [4] determines the threshold based on a radial basis function neural network. Since there is no explicit definition of the optimum threshold for the desired false alarm rate, the correct value is found via grid search. Another study explains an umbrella algorithm which is designed as a framework for NP classification and applicable to well-known classifiers such as Naive Bayes, SVM, random forest and logistic regression [3]. In this approach, the dataset is divided into two disjoint sets such that one of them is used to train the base classifier (for example SVM) and the other one is used to determine the optimum threshold for the user-specified false alarm rate. Even though this approach is capable of converting any model into an NP classifier, it requires batch processing. Hence, it is not scalable with the size of data. In the cost sensitive learning method, the model (e.g., SVM) is trained based on a loss that is obtained by assigning asymmetrical costs to the error types (false alarm vs misdetection) [6], [7]. One important drawback of these models is the predetermination of the class costs, which is often difficult to set. On the other hand, an equivalent but more intuitive setting can be obtained by requesting (from the algorithm training) a bearable false alarm level instead of trying to guess the corresponding class costs (and modifying the training accordingly). In the constrained optimization, the approach is more in line with the Neyman-Pearson formulation, where the asymmetrical class costs are parameterized by the Lagrange multiplier. In the online instances of this constrained optimization approach [8], the designed models are only capable of learning linear decision boundaries and therefore do not perform well with the nonlinear data. Even though the authors of [8] mention nonlinear extensions of the algorithm, it is left as future work.

In this paper, we introduce, as the first time in the literature to our best knowledge, an online kernel based Neyman-Pearson (NP) classification algorithm, which sequentially learns any nonlinear classification boundary by maximizing the detection power about a user-specified false alarm rate. The data processing in our algorithm is truly online through the stochastic gradient descent (SGD) updates. We solve the nonlinear NP classification problem by using a specific kernel mapping that is appropriate for computationally efficient large scale data processing. Our proposed algorithm works in a truly online framework and hence does not need to store the historical data. For this, we use the kernel approximation, where we project the received data point into a higher dimensional space by using random Fourier features [9], [10]. Then, our algorithm trains a single layer linear perceptron in the kernel space, resulting in an effective -as also experimentally demonstrated- modeling of the nonlinear decision boundaries. Moreover, our approach is scalable to large datasets through a parameter of the approximated kernel size, which provides a control over computing resources.

II. Problem description

We study the online binary classification problem in the Neyman-Pearson testing framework. We sequentially observe the data $\boldsymbol{x}_t \in R^d$ and decide about its label $\hat{y}_t \in \{1, -1\}$ as $\hat{y}_t = 1$ if our model f_t at time t provides $f_t(\boldsymbol{x}_t) \geq 0$, and as $\hat{y}_t = -1$, otherwise. Then, we update our model f_t to obtain f_{t+1} based on the error $y_t - \hat{y}_t$ and discard the observed data, i.e., $\boldsymbol{x}_t$ and y_t, without storing. [1] During this truly online process, we directly and sequentially implement the Neyman-Pearson (NP) optimization. In particular, the nondetection rate $\mathrm{P}_{\text{nd}} = \mathrm{P}(\hat{y}_t = -1 | y_t = 1)$ is minimized by keeping the false alarm rate $\mathrm{P}_{\text{fa}} = \mathrm{P}(\hat{y}_t = 1 | y_t = -1)$ under a user-specified threshold. Hence, we solve the following Neyman-Pearson optimization in an online framework

$$\min \mathrm{P}_{\text{nd}} \text{ subject to } \mathrm{P}_{\text{fa}} \leq \tau, \qquad (1)$$

where τ is the user-specified threshold upper bounding the false alarm rate.

In order to learn nonlinear classification boundaries, we explicitly transform the observation space into a high dimensional kernel space via the transformation $\phi_{\boldsymbol{\alpha}} : R^d \to R^{2D}$ such that $R^d \ni \boldsymbol{x}_t \to \tilde{\boldsymbol{x}}_t = \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_t) \in R^{2D}$, which is based on the random Fourier features [9], [10]. Hence, we represent nonlinear model in the original data space as a linear model in the kernel space, where nonlinear classification problem becomes solvable [9]–[11]. We then use a certain variant of the perceptron algorithm [12] with the sigmoid loss in the kernel space while directly solving for the NP optimization in (1) to achieve nonlinear class separation under NP constraints, i.e., maximum detection power at a specified false alarm rate. To this end, the classification model is defined linearly in the kernel space as $f_t(\tilde{\boldsymbol{x}}_t) = (\langle \boldsymbol{w}_t, \tilde{\boldsymbol{x}}_t \rangle + b_t)$ where $\boldsymbol{w}_t \in \mathbb{R}^{2D}$ is the normal vector to the linear separator and $b_t \in \mathbb{R}$ is the bias. Thus, the decision of our classifier at time t is $\hat{y}_t = \mathrm{sgn}(f_t(\tilde{\boldsymbol{x}}_t))$. Then, our goal is to sequentially (in a truly online manner) and jointly learn all of the classifier parameters $\boldsymbol{w}_t, b_t$ under the user-specified NP constraints in (1).

[1]In this paper, all vectors are column vectors, and they are denoted by boldface lower case letters. For a vector $\boldsymbol{w}$, its transpose is represented as $\boldsymbol{w}'$, and the time index is given as subscript. Also, $1_{\{\cdot\}}$ is the indicator function returning 1 if its argument condition holds, and returning 0 otherwise.

III. Method

We construct the transformation $\phi_{\boldsymbol{\alpha}} : R^d \to R^{2D}$ such that $R^d \ni \boldsymbol{x}_t \to \tilde{\boldsymbol{x}}_t = \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_t) \in R^{2D}$ based on the fact (as provided in [9], [10]) that any continuous shift invariant kernel can be approximated as $k(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq k(\boldsymbol{x}_i - \boldsymbol{x}_j) \approx \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_i)' \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_j)$ with an appropriately randomized feature mapping function $\phi_{\boldsymbol{\alpha}}$. Note that the kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ encodes the inner product in the targeted kernel space, which is explicitly constructed here with inner products that well-approximate the kernel. Hence, we can learn the nonlinear models with linear techniques after the transformation. In our method, we use the symmetric and shift invariant rbf kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-g||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$ with the bandwidth parameter g (that is inversely related to the actual bandwidth). Then, $k(\boldsymbol{x}_i - \boldsymbol{x}_j) = E_{\boldsymbol{\alpha}}[r_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) r_{\boldsymbol{\alpha}}(\boldsymbol{x}_j)']$ (due to Bochner's theorem as provided in [9]) where the Fourier feature $r_{\boldsymbol{\alpha}}$ is $r_{\boldsymbol{\alpha}}(\boldsymbol{x}) = [\cos(\boldsymbol{\alpha}'\boldsymbol{x}), \sin(\boldsymbol{\alpha}'\boldsymbol{x})]$ and $\boldsymbol{\alpha}$ is sampled from the multivariate Gaussian distribution $p(\boldsymbol{\alpha}) = N(\boldsymbol{0}, 2g\boldsymbol{I})$ (which is the Fourier transform of the kernel in hand). Hence, we define our mapped instance as

$$\tilde{\boldsymbol{x}}_t = \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_t) = \sqrt{\frac{1}{D}} [r_{\boldsymbol{\alpha}_1}(\boldsymbol{x}_t), r_{\boldsymbol{\alpha}_2}(\boldsymbol{x}_t), \cdots, r_{\boldsymbol{\alpha}_D}(\boldsymbol{x}_t)]'.$$

After this transformation, the linear and online NP classifier [8] is implemented with differences in estimating the observed false alarm rates in the run-time. To be more precise, the NP objective is formulated as

$$\min \frac{\lambda}{2} ||\boldsymbol{w}_t||^2 + \hat{\mathrm{P}}_{\text{nd}}(f_t) \text{ subject to } \hat{\mathrm{P}}_{\text{fa}}(f_t) \leq \tau, \qquad (2)$$

where λ is the regularization parameter and the estimated nondetection and false alarm rates at time t are

$$\hat{\mathrm{P}}_{\text{nd}}(f_t) = \frac{1}{n_{t_+}} \sum_{1 \leq t' \leq t : y_{t'} = 1} l\Big(y_{t'} f_{t'}(\tilde{\boldsymbol{x}}_{t'})\Big), \qquad (3)$$

$$\hat{\mathrm{P}}_{\text{fa}}(f_t) = \frac{1}{n_{t_-}} \sum_{1 \leq t' \leq t : y_{t'} = -1} l\Big(y_{t'} f_{t'}(\tilde{\boldsymbol{x}}_{t'})\Big), \qquad (4)$$

with $n_{t_+} = \sum_{t'=1}^{t} 1_{\{y_{t'} = +1\}}$ and $n_{t_-} = \sum_{t'=1}^{t} 1_{\{y_{t'} = -1\}}$. Note that any appropriate function can be used here to obtain a differentiable surrogate for the $0-1$ errors in estimating the error rates. However, our results in the rest of this paper are based on the sigmoid $l(z) = \frac{1}{1+\exp(hz)}$ with $h = -1$.

To solve the problem in (2), the following Lagrangian

$$L(f_t, \gamma_t) = \frac{\lambda}{2} ||\boldsymbol{w}_t||^2 + \hat{\mathrm{P}}_{\text{nd}}(f_t) + \gamma_t (\hat{\mathrm{P}}_{\text{fa}}(f_t) - \tau) \qquad (5)$$

is defined, where τ is the desired false alarm rate. Since the saddle points of (5) correspond to the local minimum of (2), the Uzawa approach [13] is used to learn the parameters in an online setting with stochastic updates. Uzawa method computes f for a specific Lagrange multiplier λ and then updates the multiplier accordingly based on the deviation of the observed the false alarm rate from the desired value. For this, the definitions in (3) and (4) are inserted into (5) to obtain the overall loss of the model as follows

$$L(f_t, \gamma_t) = \frac{1}{t}\sum_{t'=1}^{t}\left(\frac{\lambda}{2}||\boldsymbol{w}_{t'}||^2 + \mu_{t'} l\Big(y_{t'} f_{t'}(\tilde{\boldsymbol{x}}_{t'})\Big) - \gamma\tau\right), \quad (6)$$

where $\mu_{t'} = \frac{t}{n_{t_+}}$ if $y_{t'} = +1$, and $\gamma\frac{t}{n_{t_-}}$ if $y_{t'} = -1$.

In order to minimize the loss (6) we need to update the dependent variables, namely $\boldsymbol{w}_t$, b_t, which are the parameters of the perceptron defined in the higher dimensional kernel space, and γ_t, which is the asymmetric error cost corresponding to the false positive rate constraint in (1). The SGD updates for these parameters are given as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\Big(\lambda\boldsymbol{w}_t + \mu_t\nabla_{\boldsymbol{w}} l\big(y_t f_t(\tilde{\boldsymbol{x}}_t)\big)\Big), \quad (7)$$

$$b_{t+1} = b_t - \eta\Big(\mu_t\nabla_b l\big(y_t f_t(\tilde{\boldsymbol{x}}_t)\big)\Big), \quad (8)$$

$$\gamma_{t+1} = \gamma_t\bigg(1 + \beta\Big(l\big(y_t f_t(\tilde{\boldsymbol{x}}_t)\big) - \tau\Big)\bigg), \quad (9)$$

where η is the learning rate, β is the Uzawa gain [13] and τ is the user-specified threshold. Using the sigmoid loss $l(z) = \frac{1}{1+\exp(z)}$ with $z = y_t f_t(\tilde{\boldsymbol{x}}_t)$ yields the derivatives as

$$\nabla_{\boldsymbol{w}} l\big(y_t f_t(\tilde{\boldsymbol{x}}_t)\big) = -\big(1+\exp(z)\big)^{-2}\exp(z)(y_t\tilde{\boldsymbol{x}}_t), \quad (10)$$

$$\nabla_b l\big(y_t f_t(\tilde{\boldsymbol{x}}_t)\big) = -\big(1+\exp(z)\big)^{-2}\exp(h)(y_t). \quad (11)$$

Note that in our experiments, we use the following iteration for γ that we obtain directly from (5)

$$\gamma_{t+1} = \gamma_t\bigg(1 + \beta\Big(\hat{\mathrm{P}}_{\mathrm{fa}}(f) - \tau\Big)\bigg), \quad (12)$$

and we estimate the false alarm rate by counting the errors in the sliding window (W_s) for the negative class to obtain a more robust estimate (different from [8]) instead of accumulating the corresponding losses. Size of the sliding window is selected as $10/\tau$ where τ is the user-specified false alarm rate. Based on the derivations provided above, we update our parameters at each time to achieve the maximum detection rate while keeping the false alarm rate under the user-specified threshold. Hence, we construct our kernelized method in Algorithm 1 that can be used for nonlinear classification problems.

IV. Experiments

In this section, we demonstrate the performance of our classification algorithm OKNP in comparison with a linear classifier, i.e., Online Stochastic Neyman-Pearson (OLNP) [8], on several different datasets [14], [15]. Details of the binary classification datasets are given in Table I. In our experiments, we compare the achieved P_{nd} and P_{fa} of the algorithms for the desired false alarm rates in the set $\tau \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$. For this purpose, we reserve 25% of the datasets for the test and the remaining for the training process, and we extend the training set to 15×10^4 data instances by using shuffling and concatenation to better investigate the learning curves. We run the algorithms with training instances in the online setting and evaluate P_{nd} and P_{fa} with the test instances. We repeat this process for all algorithms 15 times on each dataset and report the average achieved P_{nd} and P_{fa} along with their standard deviations. Note that we make some modifications on the implementation of OLNP [8] for a fair evaluation. We apply our iteration method for γ in (12) instead of the one proposed in [8], and we evaluate the achieved P_{nd} and P_{fa} based on the test parts described above as in our method. Also, note that we use the data points in the same order for all algorithms, and we standardize (zero mean, unit variance) all datasets prior to training and testing. We specify the bandwidth parameter g as $\frac{1}{d}$ for $x \in R^d$. Moreover, we select the projection space dimension D (higher dimensional kernel space) via cross validation from the set $d \times \{2, 3, 4, 5\}$ according to the Lagrange cost in (6). Here, we define Uzawa gain [13] as $\beta = \eta \times 0.01$ since β should be smaller than η to avoid increasing the priority of the false alarm rate over the detection rate.

Algorithm 1 Online Kernel Neyman-Pearson Classifier (OKNP)

```
 1: Set the desired (or target) false positive rate (TFPR) $\tau$
 2: Set the input parameters: regularization parameter $\lambda$, kernel space dimension $2D$, bandwidth $g$ for the rbf kernel
 3: Initialize $\boldsymbol{\alpha}$ by drawing $D, 1 \le j \le D$, i.i.d. samples as $\boldsymbol{\alpha} \sim N(\boldsymbol{0}, 2g\boldsymbol{I}) : \boldsymbol{\alpha} \in \mathbb{R}^d$
 4: Initialize the learning rates $\eta_1$ and $\beta_1$
 5: Initialize $\boldsymbol{w}_1$, $b_1$, $\gamma_1$
 6: Initialize $n_{t_+} = n_{t_-} = 0$, and sliding window size $W_s = \frac{10}{\tau}$.
 7: for $t = 1, 2, \dots$ do
 8:     Receive $\boldsymbol{x}_t$
 9:     Calculate $\tilde{\boldsymbol{x}}_t = \phi_{\boldsymbol{\alpha}}(\boldsymbol{x}_t)$ and $f_t(\boldsymbol{x}_t) = \boldsymbol{w}_t'\tilde{\boldsymbol{x}}_t + b_t$
10:     Calculate the current decision as $\hat{y}_t = \mathrm{sgn}(f_t(\boldsymbol{x}_t))$
11:     Observe $y_t$
12:     Calculate $n_{t_+} = n_{t_+} + 1_{\{y_t=1\}}$ and $n_{t_-} = n_{t_-} + 1_{\{y_t=-1\}}$
13:     Calculate $\mu_t = \frac{t}{n_{t_+}}1_{\{y_t=1\}} + \frac{\gamma t}{n_{t_-}}1_{\{y_t=-1\}}$
14:     Update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t\Big(\lambda\boldsymbol{w}_t + \mu_t\nabla_{\boldsymbol{w}} l\big(y_t f_t(\boldsymbol{x}_t)\big)\Big)$, cf. (7)
15:     Update $b_{t+1} = b_t - \eta_t\Big(\mu_t\nabla_b l\big(y_t f_t(\boldsymbol{x}_t)\big)\Big)$, cf. (8)
16:     Update $\gamma_{t+1} = \gamma_t\bigg(1 + \beta_t\Big(\hat{\mathrm{P}}_{\mathrm{fa}} - \tau\Big)\bigg)$, if $y_t = -1$, cf. (12) and the corresponding explanation about the estimate $\hat{\mathrm{P}}_{\mathrm{fa}}$ of the false positive rate based on a sliding window approach
17:     Update $\eta_{t+1} = \eta_1(1 + \lambda t)^{-1}$
18:     Update $\beta_{t+1} = \beta_1(1 + \lambda t)^{-1}$
19: end for
```

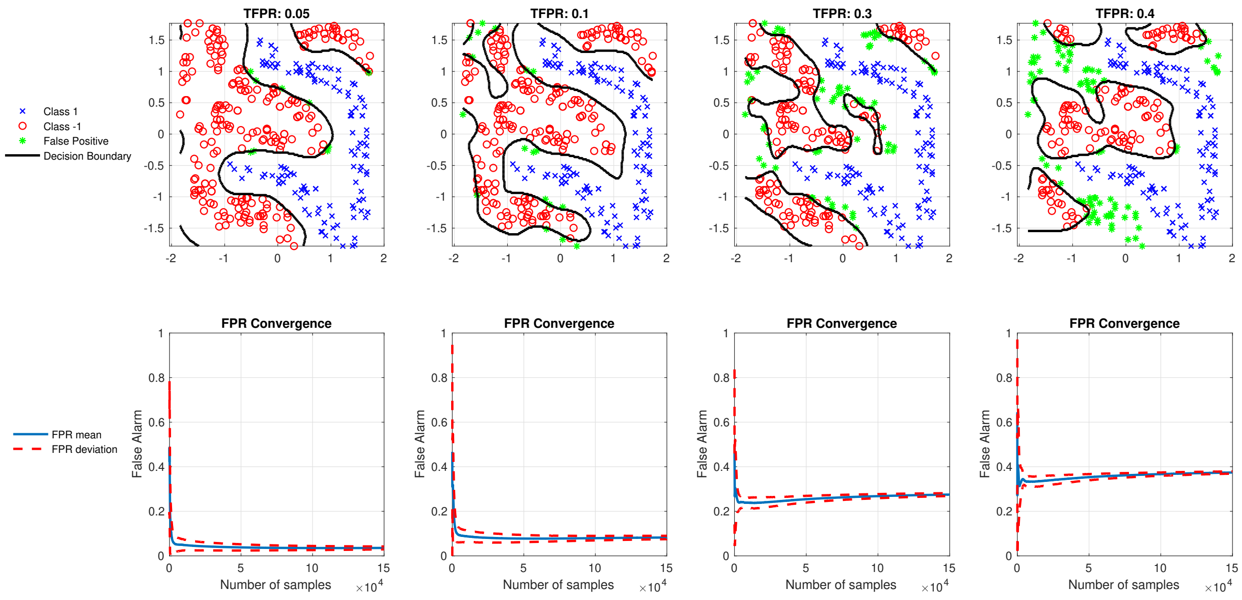

Fig. 1: The decision boundaries (upper) and false positive convergence behaviour (lower) are shown for the fourclass dataset. The straight and dashed lines represent the means and standard deviations of the results of experiments conducted on 15 random permutations of the same data (fourclass), respectively.

We provide the decision boundaries and the false alarm convergence graphs of OKNP in Fig. 1 for $P_{fa} = \{0.05, 0.1, 0.3, 0.4\}$. Here, the upper four graphs represent the change of the decision boundaries as the target false positive rate (TFPR) increases. For large false alarms, the model allows more false positives in order to meet the NP constraints. Moreover, the bottom graphs represent the false alarm convergence behaviour of the model. Initially, the standard deviation (dashed lines) is high due to the low number of learning samples processed by the model. As more samples are observed by the algorithm, the standard deviation in both false positive rate (FPR) and true positive rate (TPR) decreases, and the model approaches to the desired false alarm rate.

For further analysis, we run all algorithms 15 times on spiral, fourclass, banana, pageblocks, pendigits and gammatel datasets in Table I and record the achieved false positive rate (FPR) and true positive rate (TPR) values for the target false alarm rates (TFPR) $\tau \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$. Note that the labelled datasets in Table I have linear properties, and in the experiments, both linear and nonlinear datasets are used to fully test the performance of our algorithm. Out of 15 runs, 10 results with low Lagrange cost as in (6) are selected in order to eliminate runs with bad convergence behaviour. We generate the receiver operating characteristics (ROC) curves by using the filtered FPR and TPR as in Fig. 2, where the marker locations are calculated as the mean of 10 runs and the error bars are the standard deviation of the same set for all algorithms. Since both FPR and TPR are calculated at each run, error bars are given in both dimensions. The ROC

Datasets	Features	n_-	n_+
Spiral	2	211	101
Fourclass	4	555	307
Banana	2	2924	2376
Pageblocks*	10	4913	560
Pendigits*	16	6714	780
Gammatel*	10	12332	6688
Cod-rna	8	325710	162855
Covertype*	54	560502	20510

TABLE I: Datasets used in the experiments, where the marked ones (with "*") seems to have linear separations.

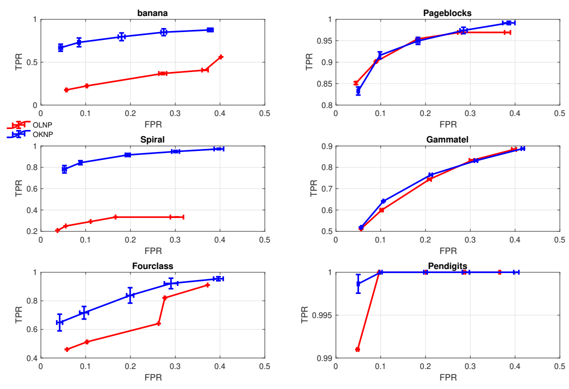

Fig. 2: ROC curves of OLNP and OKNP for small datasets. Markers points are calculated as the mean of 10 different runs and the errorbars represent deviations in FPR and TPR.

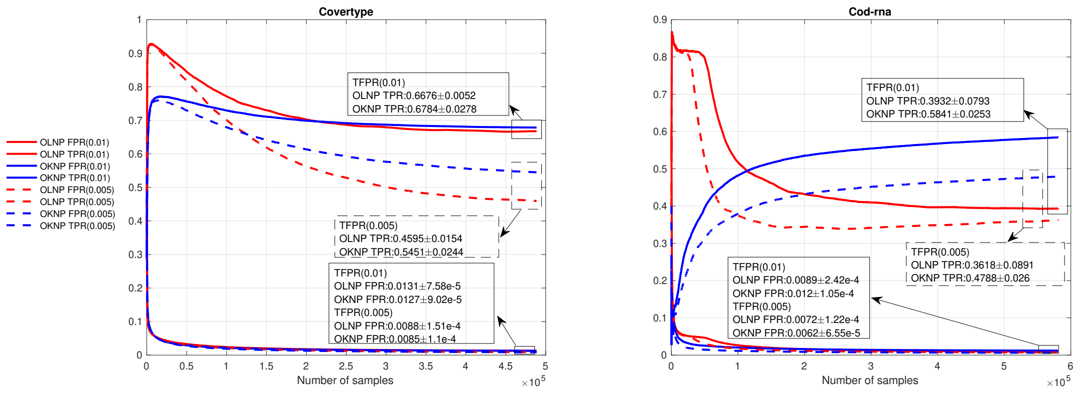

Fig. 3: TPR and FPR convergence graphs of OLNP and OKNP for $\tau \in \{0.005, 0.01\}$.

comparisons of OKNP and OLNP given in Fig. 2 shows that our proposed model OKNP significantly outperforms OLNP with its nonlinear modeling capabilities particularly in nonlinear datasets such as banana, spiral and fourclass. In the case of datasets with linear properties, i.e., pageblocks, pendigits and gammatel, our algorithm OKNP still performs well even though it learns higher number of parameters of a more complex model compared to OLNP, which is linear by design. Hence, the proposed algorithm OKNP can be used in both types of datasets with high classification performance due to its high modelling power, which makes it superior particularly in nonlinear datasets.

Moreover, we perform experiments on large datasets to investigate the performance of our model with high number of samples under harder NP constraints. For this, we use cod-rna and covertype datasets in Table I, where we update the NP constraints with smaller TFPR $\tau \in \{0.005, 0.01\}$. It is important to note that since our algorithm is truly online, the necessary computational power increases linearly with the sample size. Therefore, it is highly scalable to the big data compared to batch algorithms. The TPR and FPR convergences of OKNP and OLNP are given in Fig. 3. In these experiments, both algorithms successfully converge to the user-determined false alarm rates. However, our algorithm OKNP outperforms OLNP in terms of TPR while ensuring strong FPR controllability.

V. Results

We introduced an online nonlinear Neyman-Pearson classification method to maximize the detection power while bounding the false alarm rate under a user-specified threshold for any nonlinear dataset. For this, we map the observed data to a high dimensional kernel space through a mapping function whose parameters are randomly initialized. We then decide its label with a linear classifier in the kernel space. Since the classifier parameters are learned sequentially through stochastic gradient descent updates, our algorithm works in an online framework. As a result of our experiments, we observe significant performance improvement achieved by our method with respect to the online linear NP classifier proposed in [8] in terms of detection power and false alarm controllability.

References

[1] X. Tong, Y. Feng, and A. Zhao, "A survey on Neyman-Pearson classification and suggestions for future research," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 8, no. 2, pp. 64–81, 2016.

[2] A. Cannon, J. Howse, D. Hush, and C. Scovel, "Learning with the Neyman-Pearson and min-max criteria," *Los Alamos National Laboratory, Tech. Rep. LA-UR*, pp. 02–2951, 2002.

[3] H. V. Poor, *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[4] D. Casasent and X.-w. Chen, "Radial basis function neural networks for nonlinear Fisher discrimination and Neyman–Pearson classification," *Neural Networks*, vol. 16, no. 5-6, pp. 529–535, 2003.

[5] X. Tong, Y. Feng, and J. J. Li, "Neyman-Pearson classification algorithms and NP receiver operating characteristics," *Science advances*, vol. 4, no. 2, p. eaao1659, 2018.

[6] M. A. Davenport, R. G. Baraniuk, and C. D. Scott, "Tuning support vector machines for minimax and Neyman-Pearson classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 10, pp. 1888–1898, 2010.

[7] X.-Y. Liu and Z.-H. Zhou, "Learning with cost intervals," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 403–412.

[8] G. Gasso, A. Pappaioannou, M. Spivak, and L. Bottou, "Batch and online learning algorithms for nonconvex Neyman-Pearson classification," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–28, 2011.

[9] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[10] F. Porikli and H. Ozkan, "Data driven frequency mapping for computationally scalable object detection," in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2011, pp. 30–35.

[11] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[12] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[13] H. Uzawa, "Iterative methods for concave programming," *Studies in linear and nonlinear programming*, vol. 6, pp. 154–165, 1958.

[14] D. Dua and C. Graff, "UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences. 2019."

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.